Bringing Multigrid to the Masses

Robert D. Falgout
Lawrence Livermore National Laboratory

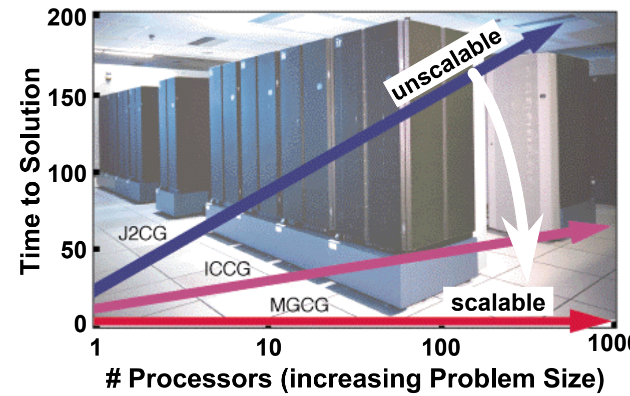
SciDAC Review
May 13, 2003

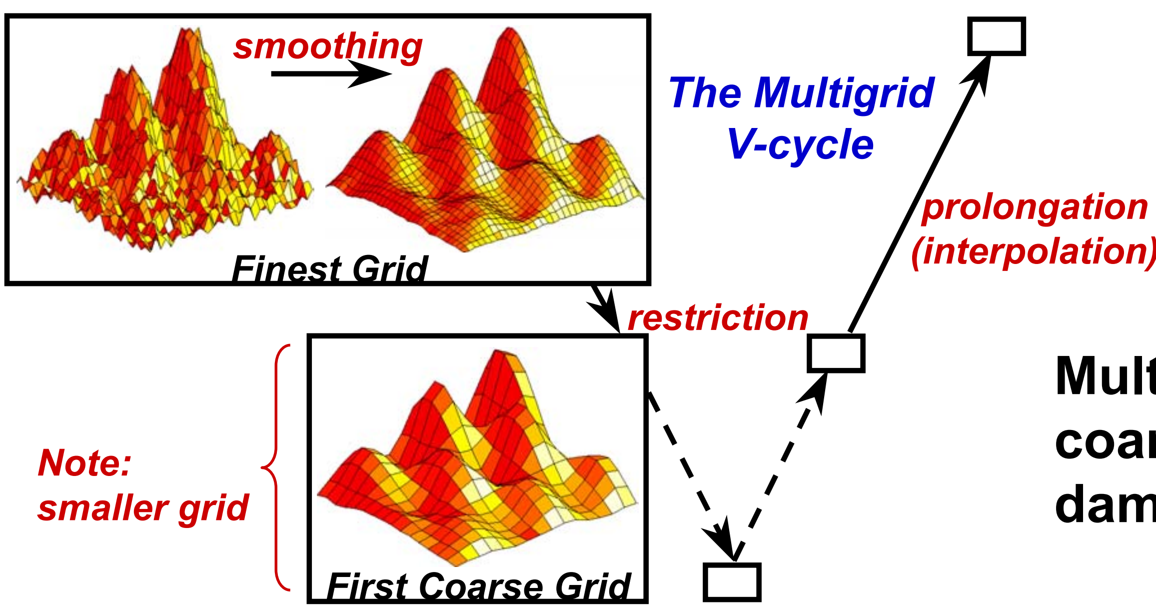
SciDAC Scientific Discovery through Advanced Computing

TOPS Terascale Optimal PDE Simulations

Multigrid is Optimal: convergence rate is independent of discretization parameters

Want (nearly) constant solution time as problem size grows in proportion to the number of processors

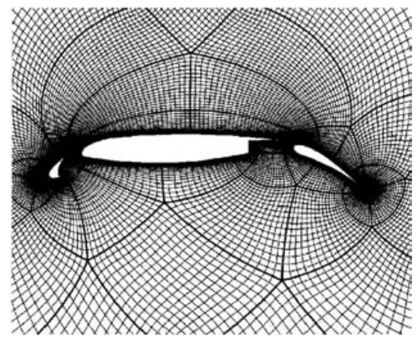


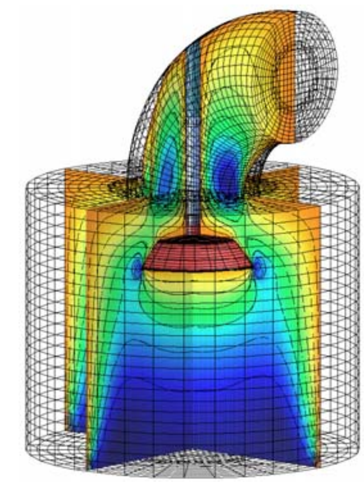


Multigrid methods use coarse grids to efficiently damp out smooth error

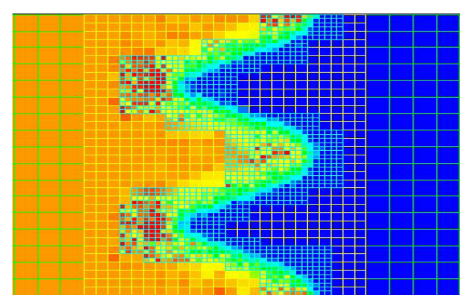
We are developing geometric multigrid for semi-structured-grid problems

- **Grids are mostly—but not entirely—structured**
- **Examples:** block-structured grids, structured adaptive mesh refinement (AMR) grids, overset grids





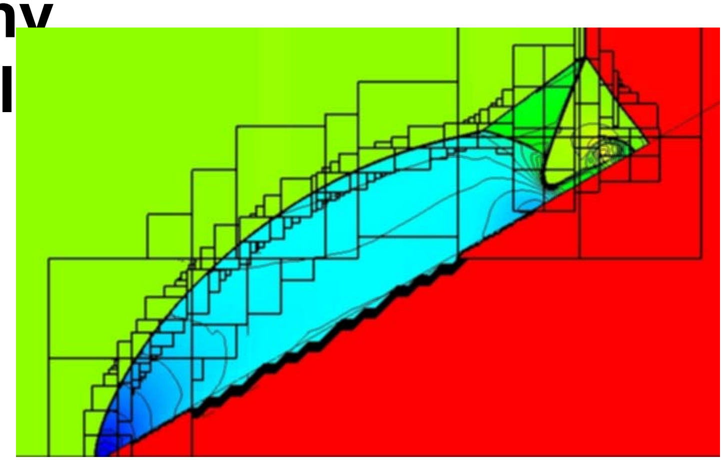
- **Basic idea: exploit grid structure where present**



- **Focusing now on solvers for AMR (for APDEC)**

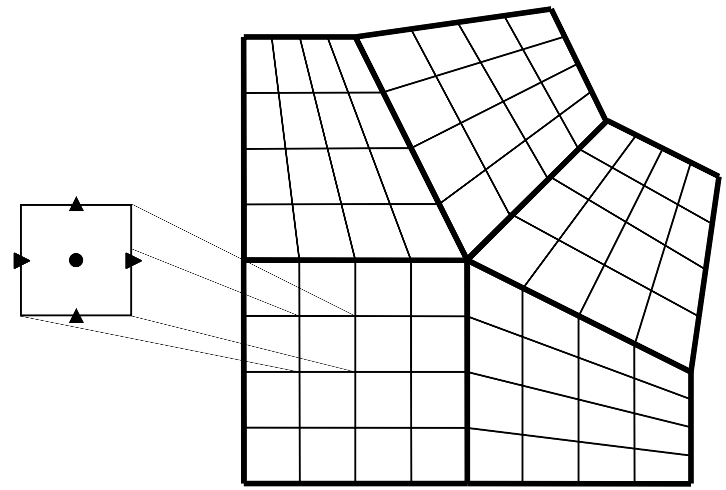
We are developing parallel Fast Adaptive Composite Grid (FAC) methods for AMR

- **AMR application developers have had little linear solver library support**
  - Usually have to “roll their own” solvers
  - Currently *hypre* is being used for level & bottom solves
- **FAC was designed specifically for AMR (McCormick)**
  - Utilizes the grid hierarchy
  - Involves so-called “level
- **Library design issue:**
  solvers like FAC require (additional) information about structure

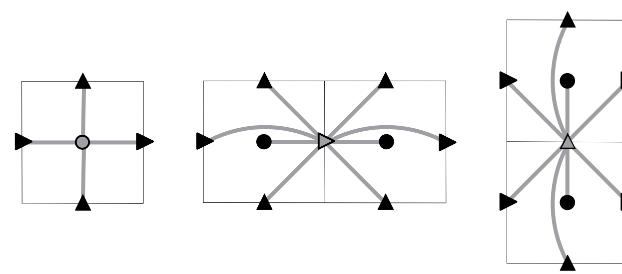


Semi-structured grid interface (SEMI) is our vehicle to deliver FAC… and more

- SEMI is one of *hypre*'s conceptual interfaces
- AMG, ILU, already available
- Currently used by ASCI for block-structured apps



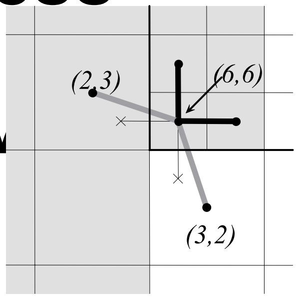
- Augmenting for AMR users
  - more natural way of handling coarse-fine boundaries
- Releasing a spec & impl
  - De facto standard
  - PETSc solver availability
  - CTS^2 component



A block-structured grid with 3 variable types and 3 discretization stencils

We are interacting closely with APDEC

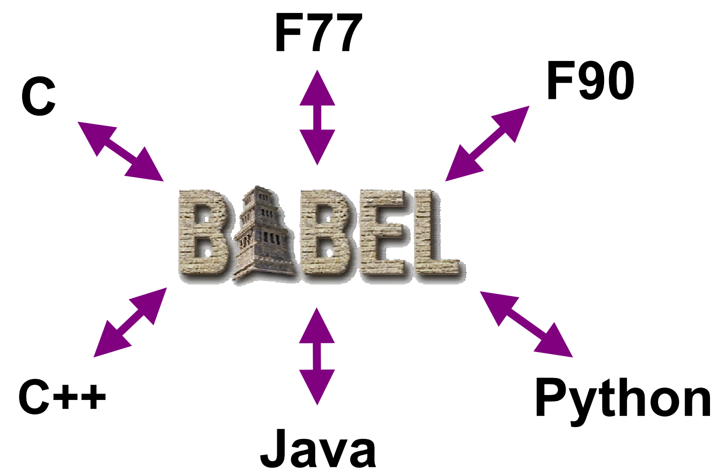
- **Integration into Chombo is in progress**
  - — ***hypre*** **level / bottom solvers available**
  - — **Some performance issues to be resolv**



- **Implementing parallel FAC code**
  - — Works in serial; finishing up for parallel
- **Developing FAC for anisotropic problems**
  - — Needed for fusion applications
- **New non-Galerkin option in PFMG for level solves**
  - — Retains discretization stencil on all MG levels
  - — Reduces storage requirements
  - — 2x speedup in 2D; expect even more in 3D
- **Constant-coefficient solvers**

First release of *hypre* featuring the Babel language interoperability tool

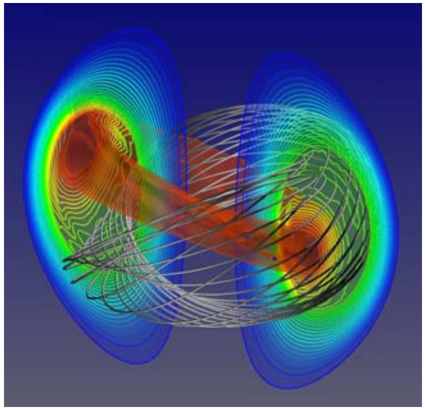
- Babel provides:
  - language interoperability
  - OO support
- Beta release 1.8.0b uses Babel for two major interface classes:
  - IJ system builders
  - ParCSR solvers (e.g., AMG)



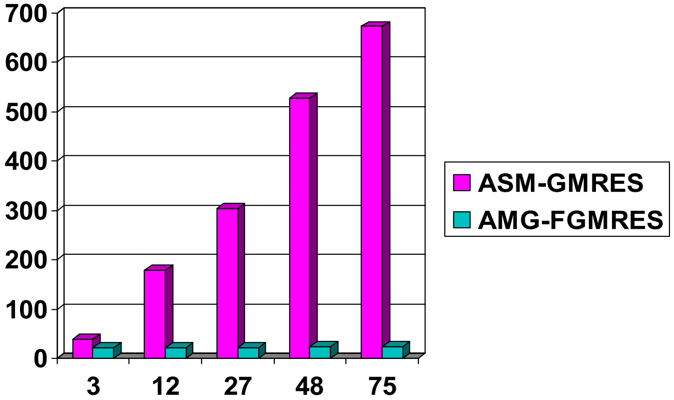
- **Long term plan:** Migrate all *hypre* interfaces to Babel to improve and expand language support
- Babel / SIDL crucial for **PETSc-*hypre* interoperability plans** and **CTS^2 interactions**

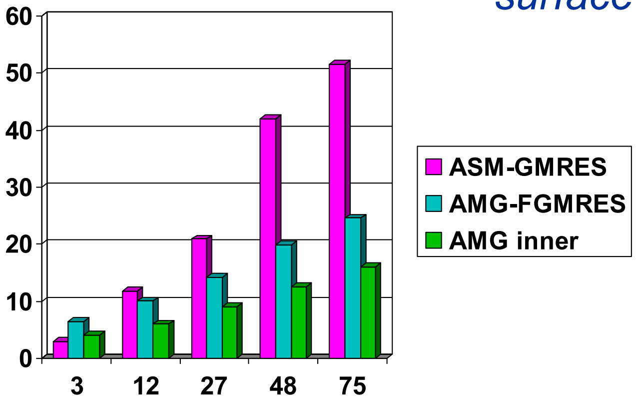
We are speeding up tokamak simulations through PETSc-*hypre* combo

- **CEMM's M3D code is built upon PETSc's distributed data structures**
- ***hypre*'s AMG solver (via PETSc) is now speeding up simulations**
  - Perfect iteration scaling
  - Still performance issues to resolve
  - Time is halved or better for large runs



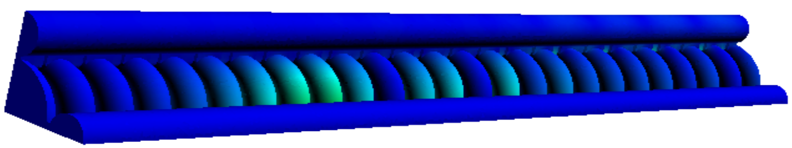
NSTX sawtooth, showing pressure contours and surface with some B-lines



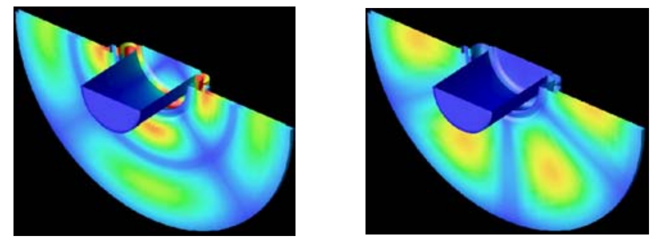


We are generalizing our *AMG* framework to address new problem classes

- **Maxwell and Helmholtz problems have huge near null spaces and require more than pointwise smoothing to achieve optimality in multigrid**



Model of a section of the Next Linear Collider structure



Resonant frequencies in a Helmholtz Application

- **Our new theory allows for any type of smoother, and also works for a variety of coarsening approaches (e.g., vertex-based, cell-based, agglomeration)**
- **A paper is in the works (will submit any day now)**

The new theory separates construction of coarse-grid correction into two parts

- **The following measures the ability of a given coarse grid Ω_c to represent algebraically smooth error:**

$$\mu^* \equiv \min_{P} \max_{e \neq 0} \mu(PR, e)$$

- **Theorem:** (1) Assume that $\mu^* \leq K$ for some constant K. (2) Assume that any one of the following holds for $\eta \geq 1$:

$$\langle A\,Qe, Qe \rangle \leq \eta \langle Ae, e \rangle, \quad \forall e$$

$$\langle A\,(I-Q)\,e, (I-Q)\,e \rangle \leq \eta \langle Ae, e \rangle, \quad \forall e$$

$$\left\langle APe_c, Se_s \right\rangle^2 \leq (1-\eta^{-1}) \left\langle APe_c, Pe_c \right\rangle \left\langle ASe_s, Se_s \right\rangle, \quad \forall\, e_c, e_s$$

  Then, $\mu(PR, e) \leq \eta K,\ \forall e$.

- **(1) insures coarse grid quality** – use *CR*
- **(2) insures interpolation quality** – necessary condition!

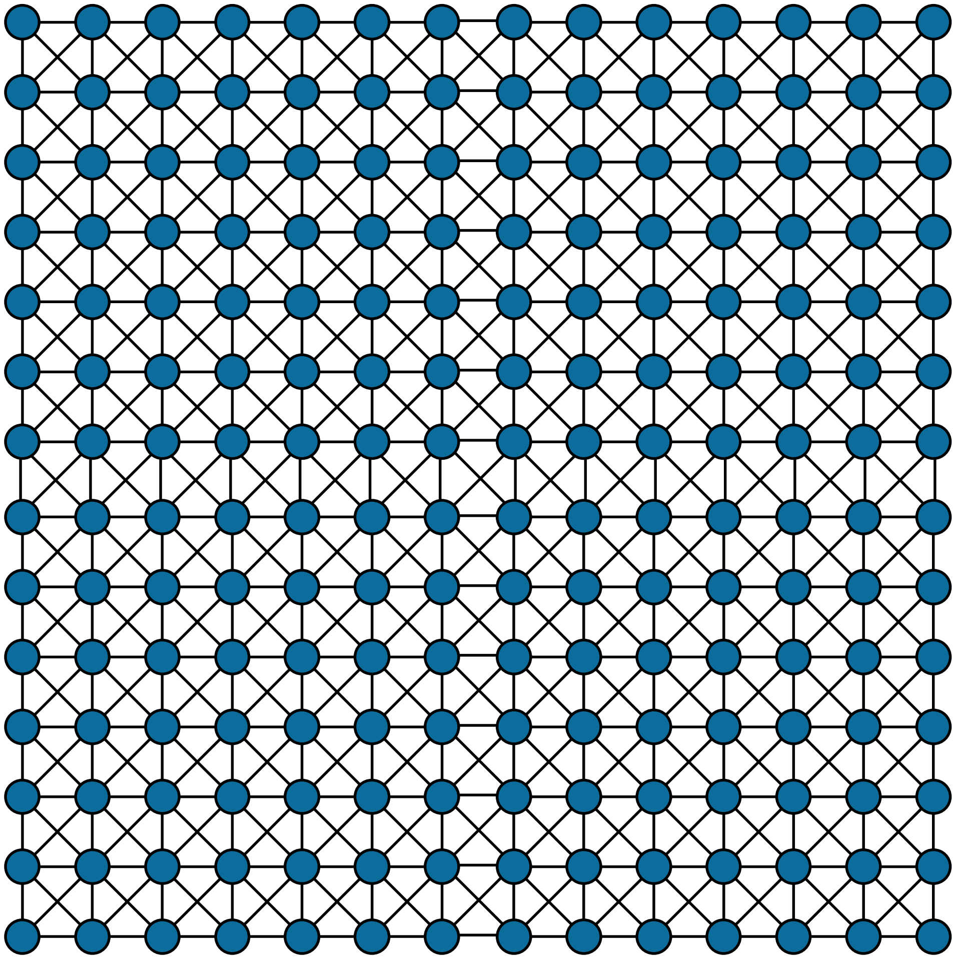
CR is an efficient method for measuring the quality of the set of coarse variables

- *CR* (Brandt, 2000) is a modified relaxation scheme that keeps the coarse-level variables, Ru, invariant
- We have defined several variants of *CR*, and shown that fast converging *CR* implies a good coarse grid:

$$\mu^* \le \left(\frac{\Delta^2}{2-\omega}\right)\frac{1}{1-\rho_{cr}}$$

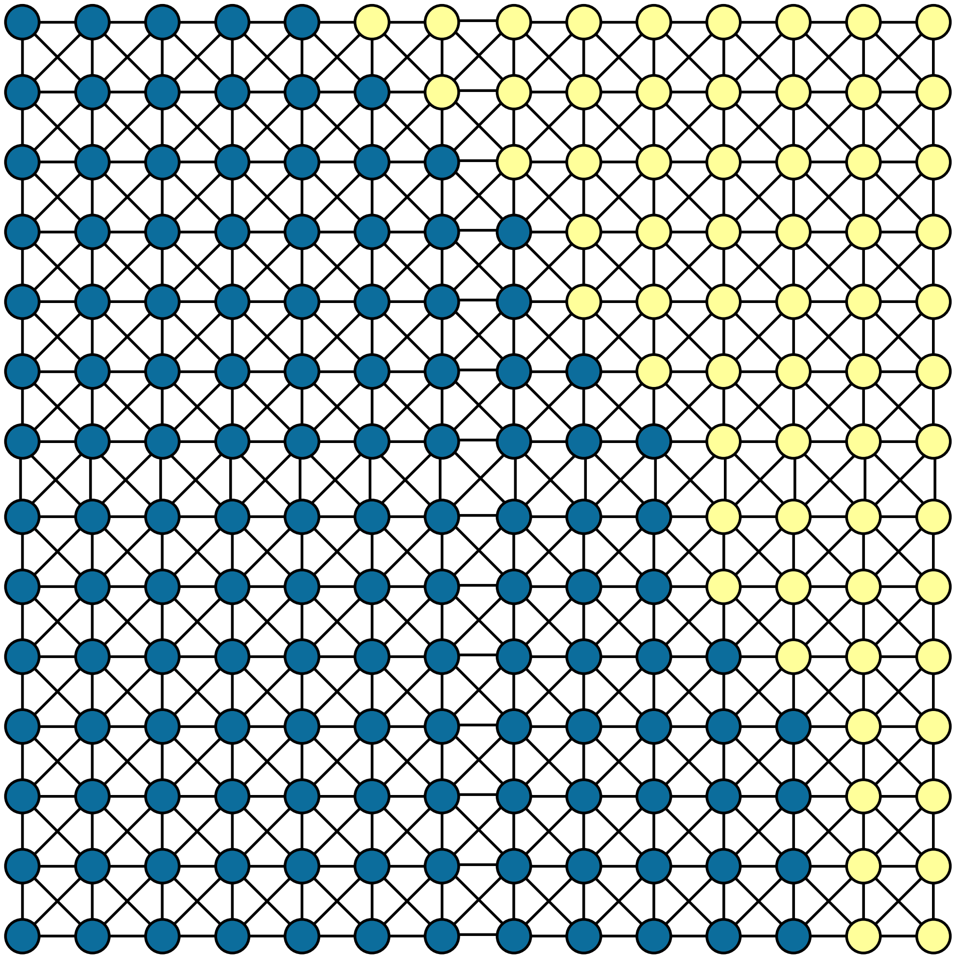
- Hence, *CR* can be used as a tool to efficiently measure the quality of a coarse grid!
- General idea: *If CR is slow to converge, either increase the size of the coarse grid or modify relaxation*
- *F*-relaxation is a specific instance of *CR*

Using *CR* to choose the coarse grid



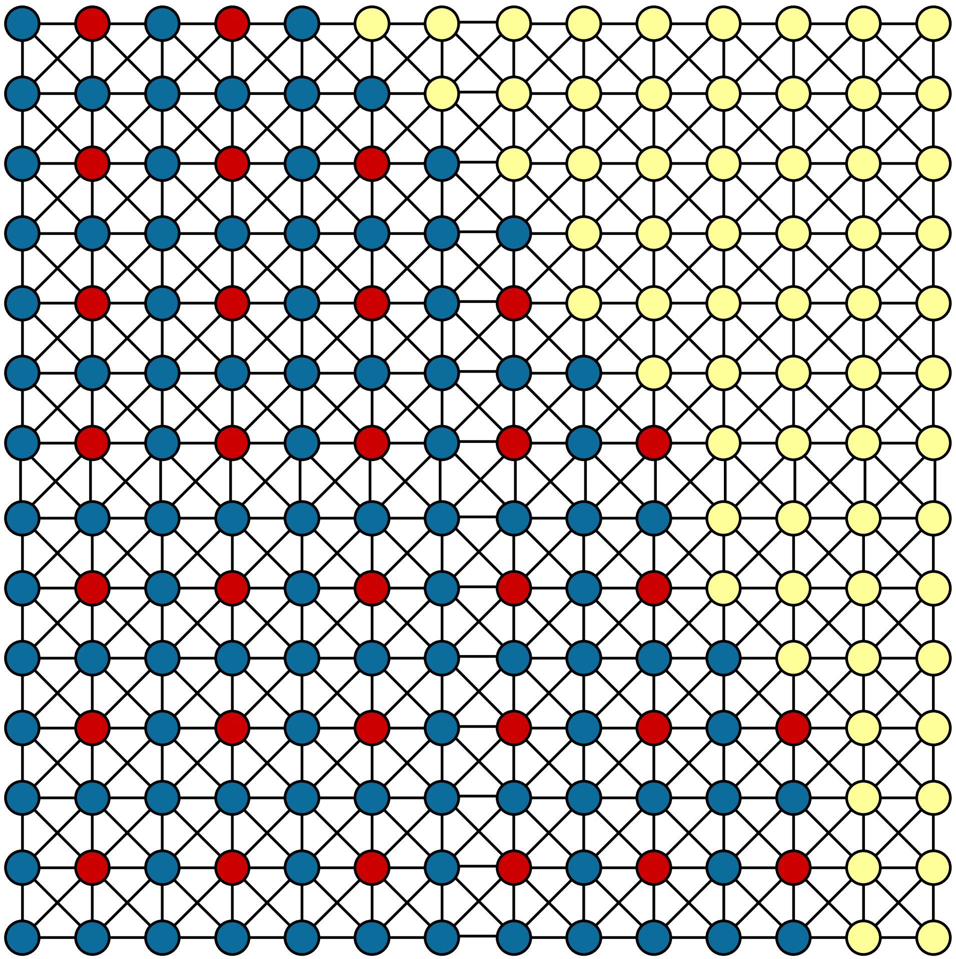
- ➔ Initialize U-pts
- ➔ Do CR and redefine U-pts as points slow to converge
- ➔ Select new C-pts as indep. set over U

Using *CR* to choose the coarse grid



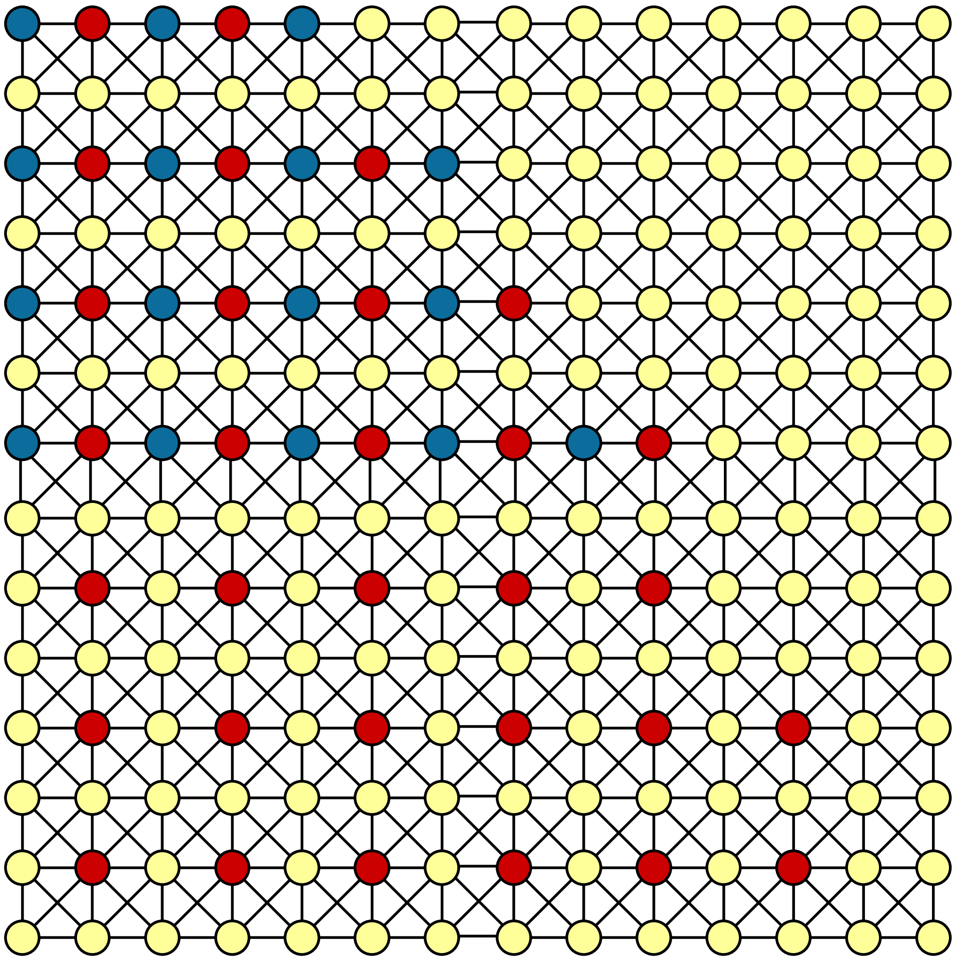
- ➔ Initialize U-pts
- ➔ **Do CR and redefine U-pts as points slow to converge**
- ➔ Select new C-pts as indep. set over U

Using *CR* to choose the coarse grid



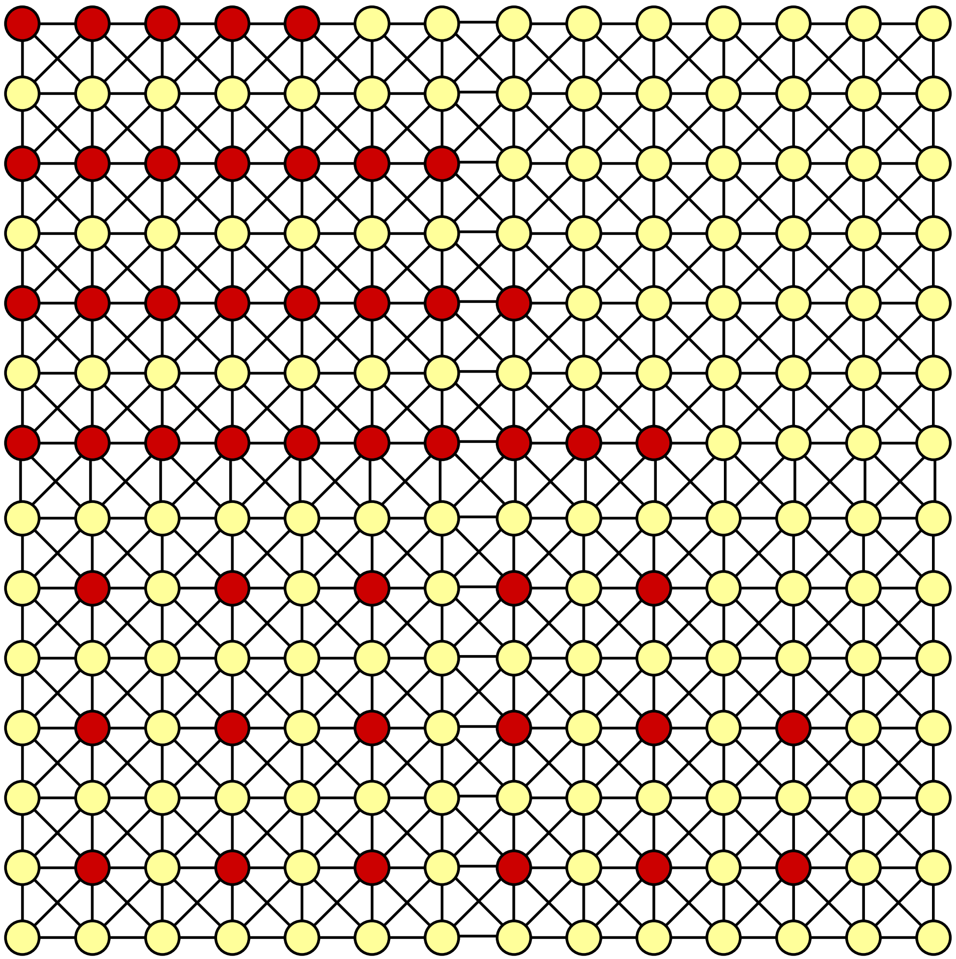
- ➔ Initialize U-pts
- ➔ Do CR and redefine U-pts as points slow to converge
- ➔ **Select new C-pts as indep. set over U**

Using *CR* to choose the coarse grid



- ➔ Initialize U-pts
- ➔ **Do CR and redefine U-pts as points slow to converge**
- ➔ Select new C-pts as indep. set over U

Using *CR* to choose the coarse grid



➔ Initialize U-pts

➔ Do CR and redefine U-pts as points slow to converge

➔ Select new C-pts as indep. set over U